Harnessing Input-Adaptive Inference for Efficient VLN

Dongwoo Kang, Akhil Perincherry, Zachary Coalson, Aiden Gabriel, Stefan Lee, Sanghyun Hong
Oregon State University
{kangdo, perincha, coalsonz, gabrieai, leestef, sanghyun.hong}@oregonstate.edu

Abstract

An emerging paradigm in vision-and-language navigation (VLN) is the use of history-aware multi-modal transformer models. Given a language instruction, these models process observation and navigation history to predict the most appropriate action for an agent. While they have significantly improved performance, the scale of these models can be a bottleneck in practical settings with limited computational resources. In this work, we propose a novel input-adaptive navigation method to enhance VLN model efficiency. We first show that existing input-adaptive mechanisms fail to reduce computations without substantial performance degradation. To address this, we introduce three adaptive algorithms, each deployed at a different level: (1) To improve spatial efficiency, we selectively process panoramic views at each observation of an agent. (2) To improve intra-model efficiency, we propose importance-based adaptive thresholding for the early-exit methods. (3) To improve temporal efficiency, we implement a caching mechanism that prevents reprocessing of views previously seen by the agent. In evaluations on seven VLN benchmarks, we demonstrate over a 2× reduction in computation across three off-the-shelf agents in both standard and continuous environments. Our code is publicly available at https://github.com/secure-ai-systems-group/adaptive-vision-and-language-navigation.

1. Introduction

Progress in vision-and-language navigation (VLN) has been enabled by larger models trained on increasingly large datasets [10, 21, 25, 29, 43]. These models can process and interpret complex data, enabling them to understand and act upon natural language instructions within visual environments. Despite the success, there is a growing concern about their computational demands. The need for substantial computational power poses a notable challenge for deployment in resource-constrained settings, such as robots, where low-power consumption becomes increasingly critical.

A potential solution to addressing these computational demands is *input-adaptive inference*. The main idea is to reduce *overthinking* [34]: as shallow networks are sufficient for the majority of samples to make decisions, e.g., class predictions, input-adaptive methods [31, 39, 51, 61] stop forwarding preemptively during inference and return intermediate outputs when the internal decisions of a model converge. During inference, they demonstrate up to 50% computational savings while preserving model performance.

In this work, we study the overthinking problem in a new domain—VLN—and propose a novel input-adaptive method to address it. Unlike prior studies on overthinking, which focus on tasks where inputs are processed independently (e.g., classification), VLN involves sequential decision-making, introducing unique problems driven by *spatio-temporal dependencies* in the inputs. Moreover, these models can be deployed for real-world navigation; thus, it is important to assess whether they are robust to common visual corruptions.

Contributions. We *first* characterize the overthinking problem in VLN by analyzing its computational bottlenecks. In our evaluation with two standard VLN agents (HAMT [10] and DUET [11]) and one continuous VLN agent (VLN-CE↻BERT [35]), we find that ∼99.5% of computations are spent in visual encoders. We also show that addressing overthinking within these visual encoders is ineffective in providing computational savings. Even with our best effort to apply the existing input-adaptive inference method, MuE [51], we demonstrate that this approach results in inaccurate navigation decisions. This increases both the time it takes for an agent to reach the target location and the overall computations while lowering the navigation success.

Second, to address this issue and achieve computational efficiency, we propose a novel input-adaptive navigation method (shown in Figure 1). We not only minimize overthinking within visual encoders, as in prior approaches, but also reduce overthinking caused by *cognitive overload* during navigation. Specifically, we focus on exploiting the spatiotemporal localities unique to VLN tasks: (1) The spatial locality: In a panorama, we find that navigable views and a few neighboring views are critical for successful navigation. We design a weighting mechanism that significantly reduces the number of views the encoder should process. We also develop an efficient subgoal module that predicts nav-

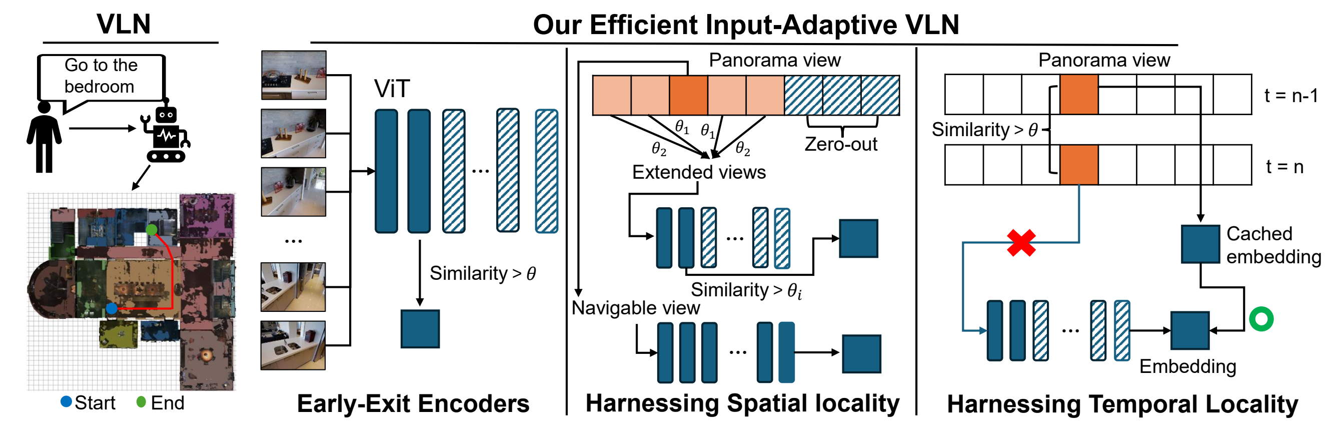


Figure 1. **Our input-adaptive, efficient navigation method.** We show on the left an agent navigating a visual environment upon a natural language instruction. On the right, we provide a high-level overview of the three input-adaptive mechanisms we propose at different levels. The shaded rectangles (embeddings) and squares (views) correspond to components that our method skips or zeroes out to improve efficiency.

igable views from laser scans, enabling compatibility with continuous environments where such views are unknown to the agent as priors. (2) The temporal locality: We find that an agent encounters identical or nearly identical views in consecutive navigation steps. We design a locality-sensitive hashing algorithm to avoid computing these matching views during navigation. (3) We lastly develop an algorithm for dynamically adapting the thresholds for an existing early-exit method based on the locality to further reduce computations.

Third, we comprehensively evaluate our input-adaptive navigation method on 7 VLN benchmarks across three popular agents. Our method reduces computations by up to 60% with an average drop in SR of 11.7% in the standard setting. In the more challenging continuous setting, it achieves ∼86% savings with an even smaller 8% SR decrease. In contrast, baseline methods experience up to 33.6% performance loss and fail to reduce computations. Our ablation study also shows how a practitioner can configure our method for their navigation environments and the factors we do not rely on. Moreover, we examine the robustness of our method to natural visual corruptions that may occur during navigation (such as lighting changes). We show that while both the baseline and our method show a slight increase in the computations, our approach loses 7–10% more performance.

2. Related Work

Vision-and-language navigation (VLN). Research in this area has been supported by the development of high-quality simulators such as Matterport3D [7] and Habitat [49], which we leverage in our work. Agents developed towards this challenging problem have ranged from earlier recurrent models [1, 20] to more recent transformer-based models [10, 11, 29, 33, 37, 46, 58]. While recent agents achieve superior performance, larger models combined with high-fidelity panoramic observation and action spaces have led to their increased complexity and higher computational costs during inference. Our work is the first providing a tunable trade-off between computational demands and accuracy.

VLN agents are studied in two environmental settings. The first is discrete (standard) VLN, where agents teleport between neighboring nodes in a known navigation graph that provides candidate navigable views as an agent's action space. To circumvent the unrealistic assumptions of navigation graphs, prior work [36] proposes continuous VLN, where agents instead use low-level actions and estimate navigable views using a sub-goal generation module. We empirically achieve large computational savings in both settings.

Input-adaptive mechanisms for computational efficiency. Prior work introduces two distinct mechanisms for input-adaptive inference: adaptive neural networks (AdNNs) and multi-exit architectures. AdNNs [19, 56] dynamically skip certain blocks of the model to save computations during inference. In contrast, multi-exit architectures [31, 34, 52, 61] introduce an additional component to the model, such as classifiers attached to each internal layer (early-exits), allowing the model to preemptively stop running forwards once stopping criteria are met. Both mechanisms demonstrate computational savings while minimizing performance loss in classification tasks (e.g., a 50% reduction in computation at a utility loss of ∼10%). We use multi-exit architectures, as AdNNs are limited to residual networks. Most multi-exit architectures are developed for classification tasks and are *not* compatible with VLN, where an agent utilizes visual and/or language representations generated from encoders. The closest work by Tang *et al.* [51] developed an adaptation (MuE) to Transformer-based encoders, but despite our best efforts, it does not provide any computational savings in VLN tasks (shown in Sec 3.1). Similarly, Yue *et al.* [62] propose an early-exit strategy for MLLM-based embodied AI, but do not address navigation tasks and focus on sequential action predictions rather than spatio-temporal dependencies in visual observations. A separate line of research explores methods for compressing models, such as quantization and pruning. These methods are orthogonal to our study and can be applied in conjunction with our method (see Appendix I).

3. Input-Adaptive Efficient VLN

3.1. Characterizing Overthinking in VLN

Computational bottleneck. The first step in designing an efficient input-adaptive mechanism is to understand the computational bottleneck of an agent during navigation. Because no prior work has studied which component consumes the most computational resources, we identify the bottleneck by analyzing the GFLOPs of each component in HAMT using the pre-trained agent on the R2R validation (unseen) set.

	ViT	BERT	H-ViT	CMT
GFLOPs (%)	99.50%	0.04%	0.07%	0.39%

Table 1. **Component-wise computational demands.** We run HAMT on the validation (unseen) set of R2R.

Table 1 summarizes our result. BERT requires the least computations (0.04%) as it is used only once at the beginning of navigation to encode the human instruction. In contrast, 99.5% of the computations come from the ViT, which must process 36 views per panorama at each navigation step. Considering that the remaining components, H-ViT and CMT, only account for 0.46% of the total computations, we decide to focus on the visual encoder. We note that while DUET and VLN-CE↻BERT have different architectures, the image encoder poses a similar bottleneck of at least 99.5%.

Existing mechanisms are ineffective for VLN. Next, we examine whether existing input-adaptive inference methods can provide computational savings in VLN. We find that most approaches discussed in Sec 2 are incompatible with VLN settings because they are designed for classification tasks and not encoder models. Tang *et al.* [51] proposes an input-adaptive strategy, MuE, tailored for encoder models. MuE measures the cosine similarity between the output activations from two consecutive transformer layers to determine when to stop a forward pass. If the cosine similarity becomes greater than a predefined threshold, MuE stops forwarding and skips subsequent layers. We test the MuE strategy on the ViT model in HAMT and evaluate the performance and GFLOPs of the agent on the validation (unseen) set of R2R.

Results. Table 2 shows our results for the original and MuE-based HAMT agents. With an early-exit threshold of 0.998 (optimized for the best performance-efficiency trade-off, see Appendix B), MuE reduces GFLOPs by 7% but significantly degrades performance (up to 40%). The average GFLOPs per step with MuE is 406.10, compared to 607.06 in the baseline. However, despite the significant per step GFLOPs savings, the total GFLOPs per trajectory increases because the MuE agent takes more steps to complete each trajectory.

Method	**Performance**				**GFLOPs(↓)**
	TL(↓)	**OSR(↑)**	**SR(↑)**	**SPL(↑)**	
Base	11.53	74.29	66.16	61.49	4763.24
MuE	17.37	62.20	43.93	36.92	4409.62

Table 2. **Performance and computational savings in HAMT with MuE.** Our adaptation of MuE leads to only marginal computational savings at the cost of significant performance degradation.

In Figure 2, we analyze the factors contributing to the performance loss and the limited computational savings. The left figure compares the trajectories of the original HAMT agent and HAMT with MuE. Both agents navigate to the same position until $t = 2$. At $t = 3$, the original HAMT agent correctly identifies the bathroom (green circle, top-right) and navigates to its front. However, the MuE agent only takes a small step forward and then continues to make incorrect steps until reaching the step limit. For MuE, processing fewer transformer layers led to an inaccurate understanding of the visual surroundings. As shown in the bottom-right figures, the bathroom remains visible across steps ($t \in [2, 10]$), yet the MuE agent fails to recognize it and makes suboptimal decisions. Appendix B provides a further discussion on why MuE fails when applied directly to VLN.

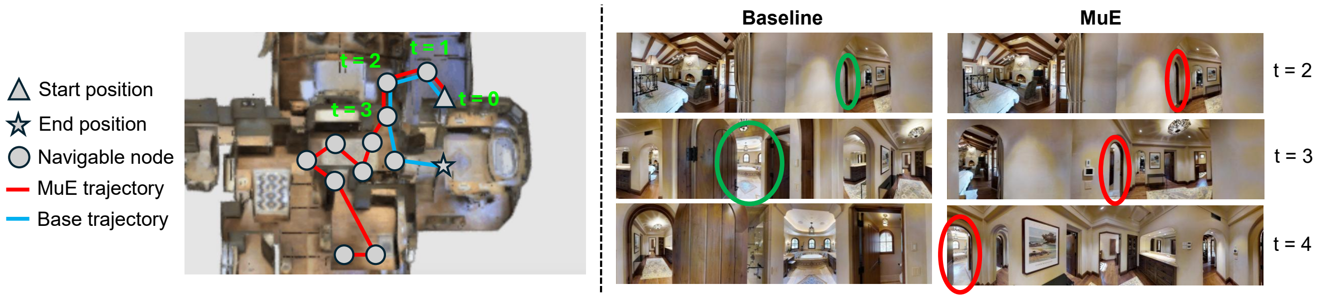
3.2. Our Methodology

Prior work on input-adaptive inference treats each input independently. As a result, existing methods inherit the *one-size-fits-all* philosophy: a model adopts a single set of configurations, such as the early-exit threshold, for all inputs. However, in dynamic settings, such as an agent navigating the physical world, inputs are not independent and depend on each other both spatially and temporally.

We introduce a novel input-adaptive inference method harnessing this unique property—spatial and temporal dependencies in the input. We first leverage spatial locality (Sec 3.2.1): among the 36 views observed by an agent at each step, we find that those close to *navigable views*—views the agent can navigate to—are important. We then propose a novel approach to assign the exit thresholds of an existing input-adaptive inference method (Sec 3.2.2) for the non-masked views to provide further computational savings. In Sec 3.2.3, we exploit temporal locality: between panorama views observed across steps, most views overlap and do not require their forward passes to be run again. Finally, while our methods are directly applicable to standard VLN agents that navigate predefined traversal graphs, we also extend them to the more practical continuous setting in Sec 3.2.4.

3.2.1. Harnessing spatial locality

Each panorama has 36 views, and the agent computes visual embeddings for each view at every navigation step. We hypothesize that only *navigable views* are crucial for navigation. Intuitively, these views form the agent's decision space,



Instruction: Go into the house and immediately go left, you should see a bathroom on your left. Go into the bathroom.

Figure 2. **Problems in employing existing input-adaptive methods in VLN.** We show that employing existing strategies leads to performance loss and an increase in computations. (Left) The increase in computations stems from inappropriate navigation actions, and (Right) such decisions come from the inaccurate understanding of the visual world, e.g., the agent confuses where to navigate.

so the information they contain should suffice for choosing the proper action. To test this hypothesis, we retain all navigable views and mask the remaining views (setting them to zero). This prevents the ViT from processing masked views, reducing computation. We evaluate the effectiveness of this approach with the HAMT agent on the validation (unseen) set of R2R. We find that it results in an 84% gain in efficiency but at the cost of a 33% reduction in SR.

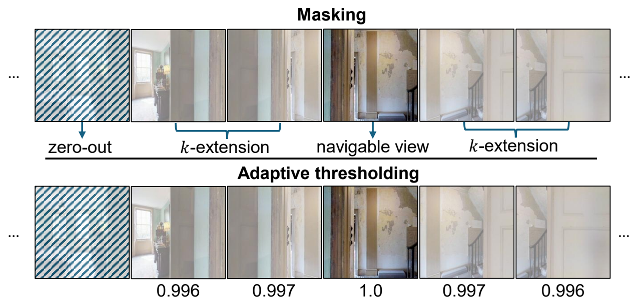


Figure 3. **Our masking and thresholding.** The top figure shows how we mask non-navigable views, and the bottom figure shows how we adaptively assign the exit thresholds of MuE.

To understand this issue, we analyze cases where masking non-navigable views prevents the agent from reaching the target. In Figure 3, processing only the navigable view (4th from the left) may obscure whether the path leads to a stairway. However, processing neighboring views increases the likelihood of correctly recognizing the path.

k-extension. To address this, we extend the number of views the agent processes near the navigable views by k. Let V be the set of n navigable views, with each v_i indexed by $\{1, 2, \dots, n\}$. The k-extension V_k^i for navigable view i is:

$$V_k^i = \{v_i^j | max(1, i-k) \leqslant j \leqslant min(i+k, 36)\},$$

where v_i^j is a non-navigable view. The union of V_k^i's gives the views to process, leaving $36 - |V_k|$ masked. With a careful calibration of k, we reduce the total computations by $2\times$ times while keeping the performance drop near 10%. In our evaluation, setting $k = 4$–6 offers the best trade-off.

3.2.2. Using adaptive thresholds as stopping criteria

On top of our k-extension, we design an adaptive mechanism to early-exit extended views and further improve the speed-up. As described in previous sections, we focus on MuE, the only early-exit mechanism compatible with encoder models.

Using budgeted-batch inference. The current implementation of MuE processes each test sample with its input-adaptive mechanism. However, this *per-sample, anytime* strategy is incompatible with our scenario, where the agent processes a batch of 36 views in a single panorama at once with the ViT. While each view in the batch should ideally exit at different layers, this per-sample approach forces all the views to use the same exit layer. To address this issue, we employ *budgeted-batch inference* [31]: each sample in a batch uses "uneven" computations, meaning that processing can stop at different layers for each sample, all within a set computational budget. We assign a sufficiently large budget so that the mechanism can handle the worst-case complexity, where none of the samples utilize early stopping.

Our adaptive thresholding. In Sec 3.2.1, we find that navigable views are most important, and a view's importance decreases with distance from navigable views. We thus design a mechanism to apply early-exit thresholds differently based on the importance of each view. We propose a concept, *rank*: low-rank views receive an aggressive (larger) threshold, while high-ranked views receive a conservative (smaller) threshold. Suppose we have a navigable view v_i at index i in a panorama and k is the number of extended views near v_i. We define the rank $R_{i,j}$ of a non-navigable view v_j relative to v_i as the difference between the indices $|j - i|$. We do not process when $R_{i,j} \geqslant k$, as views beyond k are masked. We still fully process the navigable views to retain performance. We then assign the exit threshold $T_{i,j}$ (the cosine similarity) for MuE as follows:

$$T_{i,j} = T_0 \cdot e^{(-A \cdot R_{i,j})}$$

where T_0 is the initial threshold set to 1.0, A is the aggressiveness we set to 9×10^{-4}, and $R_{i,j}$ is the rank computed above. Note that the threshold decreases as the rank increases.

3.2.3. Harnessing temporal locality

Our final insight is that during navigation, an agent will encounter similar views multiple times, leading to *temporal redundancy*. For example, views at step i are similar to those at step $i+1$. The agent may also revisit the same surroundings due to misleading navigation or encounter similar but less important surroundings, such as ceilings or walls.

To reduce temporal redundancy, we employ locality-sensitive hashing (LSH) to store and retrieve similar visual representations, avoiding redundant processing. We use SimHash [3, 8], which maps high-dimensional RGB views to low-dimensional binary encodings via random projection. Given a view v and randomly initialized hyperplanes $\{h_i\}_{i\in\{1,\dots,n\}}$, the algorithm determines which side of the hyperplane v falls on via the dot-product of v and h_i. If v is on the top side of h_i, SimHash assigns 1; otherwise, it is 0. Similar views are then encoded as the same binary encoding of length n, e.g., $010\dots1$, which we use as a key to store view-encoding pairs. Views mapped to the same key are then reused if they are sufficiently similar, which we measure using their cosine similarity. To balance performance and efficiency, we set n to 10 and the similarity threshold to 0.85 and 0.95 for standard and continuous VLN, respectively. Like early-exiting, we do not hash navigable views and fully process them. Note that with our k-extension, we limit the space complexity of caching by storing only a subset of views. With this mechanism, we achieve an additional 2–4% computational savings with minimal utility loss. See Appendix C for more details and storage overhead analysis.

3.2.4. Input-adaptive inference for continuous VLN

In continuous VLN [36], agents navigate 3D environments without predefined traversal graphs, requiring them to predict navigable views. Existing agents [30, 35] address this with *subgoal generation modules* (SGMs), which process 2D laser occupancy scans and encoded images to predict navigable views. However, this conflicts with our input-adaptive inference techniques, as the entire panorama must be processed before identifying navigable views.

To solve this problem, we introduce a *scan-only* SGM that predicts navigable views using only laser occupancy scans. We use the U-Net SGM from Krantz *et al.* [35], but remove image feature processing. Following their training procedure, we minimize the Sinkhorn divergence [14] between predictions and ground-truth subgoals. Our scan-only SGM achieves a validation loss of 0.63 on Matterport3D scene data, the same as the original work. With the ability to predict navigable views prior to image encoding, our methods become compatible with continuous VLN agents.

3.3. Putting All Together

Now, we describe how our three mechanisms are combined to perform input-adaptive inference on a panorama. We show the pseudo-code of our method in Algorithm 1:

Algorithm 1 Our Input-adaptive Navigation at Each Step

Input: a panorama P, navigable views V, visual encoder f_θ, hash table h, and the number of views to extend k
Output: a set of visual representations E for views in P

```
1:  $E \leftarrow \varnothing$
2:  for $i = 1, 2, \dots, 36$ do            ▷ Iterate over views in $P$
3:      $v_i \leftarrow P[i]$
4:      if $v_i$ in $V$ then
5:          $e_i \leftarrow f_\theta(v_i)$
6:          $E \leftarrow E + e_i$
7:      else if $i$ in $k$ proximity of any views in $V$ then
8:          $e_i \leftarrow h(v_i)$
9:          if $e_i$ does not exist then
10:             $j \leftarrow$ the index of the closest navigable view
11:             $T_i \leftarrow$ ComputeThreshold($R_{i,j}$)
12:             $e_i \leftarrow$ RunMuEInference($v_i, T_i$)
13:             $h \leftarrow$ AddToHashTable($h, v_i, e_i$)
14:         end if
15:         $E \leftarrow E + e_i$
16:     else
17:         $E \leftarrow E + \vec{0}$
18:     end if
19: end for
20: return $E$
```

(line 1-2) Initialize. It takes a panorama P and returns the visual representations of its 36 component views. We initialize the output E as empty and iterate over each view.
(line 4-6) Compute the representation of a navigable view. If the currently chosen view v_i is a navigable view, we fully compute its visual representation e_i and add it to the set E.
(line 7-15) Retrieve (or compute) the representation of the extended views. In Sec 3.2.1, to improve the visual understanding, we develop the k-extension. We process k views on both sides (left/right) of a navigable view. If v_i's representation is in the hash table h, we retrieve e_i and add it to E; otherwise, we compute e_i. Note that the hash table h is initialized at the first step of the navigation. To compute e_i, we determine v_i's rank $R_{i,j}$ and decide the exit threshold T_i. We run the inference with ViT, adapted for MuE, using T_i and store the output e_i into h and E.
(line 17) Skipping the masked view. If v_i is neither a navigable view nor in its k-extension, we store a zero-vector and move on to the next view v_{i+1}.

4. Evaluation

Datasets. Following prior work, we evaluate standard VLN with six datasets: Room-to-Room (R2R) [1], R2R-Back [10], R2R-Last [10], REVERIE [47], CVDN [53], and SOON [66]. For REVERIE, we set $k = 6$ for k-extensions to minimize performance loss while ensuring a roughly 50%

speed-up. For all other benchmarks, we achieve this using $k = 4$. To evaluate continuous VLN, we use R2R-CE [36].

VLN agents. We evaluate two off-the-shelf standard VLN agents: HAMT [10] and DUET [11]. HAMT uses a ViT [16] for vision, BERT [15] for language, and a hierarchical ViT for temporal context, predicting actions via a cross-modal Transformer. DUET also employs ViT and BERT but integrates object features (e.g., bounding boxes) and separates planning into global and local cross-modal encoders, fusing their outputs for action prediction. For continuous VLN, we use VLN-CE↻BERT [35], which uses ResNet-152 [26] for visual encoding and BERT for recurrently processing visual and language information and predicting actions.

Evaluation metrics. We evaluate navigation success using four metrics from the prior work [10, 35]: (1) Trajectory length (TL): path length of the agent in meters, (2) oracle success rate (OSR): fraction of paths with at least one viewpoint within 3 meters of the target, (3) success rate (SR): fraction of final positions within 3 meters of the target and, (4) success rate normalized by inverse path length (SPL): SR normalized by the ratio between the shortest path length and the predicted path length. For computational efficiency, we measure the GFLOPs and wall time per navigation; however, we prioritize GFLOPs as wall time depends on hardware and software implementation (see Appendix D for details).

For REVERIE and SOON, additional object features are used for navigation. We could not find the original feature extraction implementations, so we use cached object features and apply our strategy only to image feature extraction. We then report the GFLOPs for image feature processing and treat the cost of object feature extraction as a constant (C). All other benchmarks do not use object features.

4.1. Effectiveness in the Standard VLN Setting

We evaluate our method in standard VLN with two agents, six benchmarks, and five metrics described in Sec 4. We compare with two baselines: no input-adaptive methods (Base) and MuE, adapted for each agent to provide the optimal performance-efficiency trade-off. For our method, we present four variations: one with k-extension, two adding mechanisms (+LSH, +thresholds), and one combining all.

Results. Table 3 summarizes our results for REVERIE, which we prioritize because it is generally more challenging than other benchmarks [10, 11]. Due to the page limit, we show the full results for other benchmarks in Appendix D and more combinations of mechanisms in Appendix E.

For REVERIE, applying all of our mechanisms saves 49–62% computation (excluding object features) while maintaining an SR loss between 16.4–22.6%; across all benchmarks (see Appendix D), the average reduction in computations is 56% with just a 11.7% drop in SR. We set the upper limit for performance loss near 10–20% for most tasks, consistent with prior work on input-adaptive inference methods [31, 34, 39, 51, 61]. The naive adaptations of MuE only provide 21.0–24.6% computational savings and experience a significant performance drop of 41.9–56.5% in SR, as expected from our initial investigation in Sec 3.1. Our k-extension alone provides a 40.6–42.6% reduction in GFLOPs with only a 11.2–21.5% drop in SR. If we apply the adaptive thresholding (+thresholds), we achieve an additional 7.5–12.8% computational savings, with a marginal performance loss of $\sim$1–6%. Separately, combining the LSH with the k-extension results in additional computational savings up to 24.4%, with no performance loss (the SR even increases).

Agent	Method	Performance				GFLOPs(↓)
		TL(↓)	OSR(↑)	SR(↑)	SPL(↑)	
HAMT	Base	14.07	35.73	31.81	29.17	5434.71+C
	MuE	18.13	22.92	13.83	10.10	4098.77+C
	Ours (k-extension)	13.85	**26.53**	**24.96**	**22.97**	3121.20+C
	Ours (k-extension+LSH)	13.84	**26.53**	**24.96**	**22.97**	2359.72+C
	Ours (k-extension+thresholds)	13.25	26.44	24.60	22.82	2723.01+C
	Ours (All)	**13.22**	26.47	24.62	22.85	**2073.69**+C
DUET	Base	22.49	51.46	47.09	33.54	6185.15+C
	MuE	32.65	33.23	27.35	15.93	4888.35+C
	Ours (k-extension)	**21.43**	46.58	41.81	29.29	3674.29+C
	Ours (k-extension+LSH)	21.44	**46.75**	**41.95**	**29.48**	3381.45+C
	Ours (k-extension+thresholds)	22.79	44.96	39.28	27.00	3399.44+C
	Ours (All)	22.81	45.07	39.36	27.14	**3145.92**+C

Table 3. **Effectiveness of our input-adaptive inference method for standard VLN.** We show our results on REVERIE for the HAMT and DUET agents. Each cell contains the averaged metric over the trajectories in the validation (unseen) set. C is the constant cost of object feature extraction. For each metric and model, the best result across the input-adaptive methods is **bolded**.

Method	Performance				GFLOPs(↓)
	TL(↓)	OSR(↑)	SR(↑)	SPL(↑)	
Base	10.59	51.88	43.23	36.53	18074.05
SGM	9.79	46.82	39.86	34.76	2396.54
SGM+LSH	10.32	45.79	37.14	31.95	1741.33

Table 4. **Continuous VLN results.** The performance and computational savings for the baseline and our efficient VLN-CE↻BERT agents on the R2R-CE validation (unseen) set.

4.2. Effectiveness in Continuous Environments

We now study the effectiveness of our method in continuous VLN. Unlike standard agents, VLN-CE↻BERT determines actions using only navigable views. With our scan-only SGM, this allows the agent to entirely disregard non-navigable views, eliminating the need for k-extensions. Additionally, ResNet is incompatible with MuE, preventing the use of our early-exit strategy. Therefore, we evaluate two input-adaptive variants alongside the base agent: one using our scan-only SGM and the other combining it with LSH.

Results. Table 4 shows our results on R2R-CE. We first find that the baseline agent requires substantially more computations than agents in the standard VLN setting. This is primarily because viewpoints are higher resolution ($3 \times 480 \times 640$ versus $3 \times 224 \times 224$), therefore requiring more GFLOPs to process through the visual encoder. Performance is also lower, as R2R-CE is far more challenging than its discrete counterpart [35]. Despite this, our proposed techniques offer large computational savings with minimal performance drop. When applying our scan-only SGM (SGM) to VLN-CE↻BERT , we achieve an 87% reduction in GFLOPs while SR only drops by 8%. By predicting the navigable viewpoints *before* encoding them, our agent adaptively processes only the navigable views instead of the entire panorama. This results in just 5 out of 36 views being processed per navigation step, on average. Computations are reduced by ∼90% by incorporating LSH (+LSH); however, as we find in Sec 3.2.1, navigable views are more critical to navigation, so caching them leads to a larger SR drop of 14%.

4.3. Sensitivity to Our Method's Configurations

Next, we assess the sensitivity of our method's computational savings to its configurations. Our method's effectiveness depends on three key configurations: the number of extended views (k), the adaptive thresholds set based on the extension, and the similarity measure used in our LSH mechanism. Here, we show our results for R2R.

k	Performance				GFLOPs(↓)
	TL(↓)	OSR(↑)	SR(↑)	SPL(↑)	
-	11.53	74.29	66.16	61.49	4763.24
1	15.38	70.20	54.32	46.96	1250.65
2	13.67	70.84	58.19	51.99	1554.82
3	12.94	71.60	60.20	54.60	1793.76
4	12.52	71.90	61.17	55.63	2013.48
5	12.19	71.99	62.32	57.08	2216.34
6	11.89	71.99	62.84	57.94	2414.46

Table 5. **Performance and computational savings across different k values.** We evaluate with the HAMT agent in R2R.

Number of extended views k. Table 5 shows performance and GFLOPs across $k \in [1, 6]$. As k decreases, the agent processes fewer views in each panorama, yielding 49–74% computational savings at a 5–18% performance cost. Surprisingly, with $k = 1$, we save 74% of GFLOPs while only sacrificing 18% in performance (SR). We choose k such that an agent processes approximately half of the total views in each panorama; this results in $k = 4$–6 for the benchmarks we consider. Given that this strategy provides 50% computational savings across all benchmarks, even when the average number of navigable views per panorama is not used to set k, we believe the strategy is transferable to new settings.

Early-exit thresholds. We also analyze the impact of the early-exit threshold T by varying the aggressiveness factor A from 0.0 to 0.0022; the threshold decreases as a view becomes farther from a navigable view. Table 6 shows that increasing aggressiveness improves computational efficiency

A	Thresholds T				Performance				GFLOPs(↓)
	$R_{1,j}$	$R_{2,j}$	$R_{3,j}$	$R_{4,j}$	TL(↓)	OSR(↑)	SR(↑)	SPL(↑)	
0	1.0	1.0	1.0	1.0	12.52	71.90	61.17	55.63	2013.48
0.007	1.0	1.0	1.0	0.997	12.57	71.60	60.96	55.32	1973.23
0.009	1.0	1.0	0.997	0.996	12.87	71.95	60.41	54.5	1917.61
0.015	1.0	0.997	0.996	0.993	13.44	70.67	57.98	52.09	1848.89
0.022	0.997	0.996	0.993	0.990	14.61	70.29	55.60	48.56	1768.85

Table 6. **Performance and computational savings across different early-exit thresholds.** We set the aggressiveness A within [0.0, 0.022]. Note that we round the threshold to 3 decimal places and set any thresholds greater than 0.998 to 1.0 as ViTs with these thresholds will use full computations.

but reduces performance. Using $A > 0.009$ causes an SR drop of over 10%, so we set $A = 0.0009$.

Using different similarity metrics. In Sec 3.2.3, our primary metric for computing similarity between views is cosine similarity. We explore whether employing different similarity metrics can further enhance the effectiveness of our method. To evaluate this, we test four additional metrics: visual features extracted from ViT's first-layer activations, SSIM [57], FSIM [63], and LPIPS [65]. We also test SURF [5] and SIFT [41] in Appendix G, but they fail to match visually similar views in consecutive navigation steps.

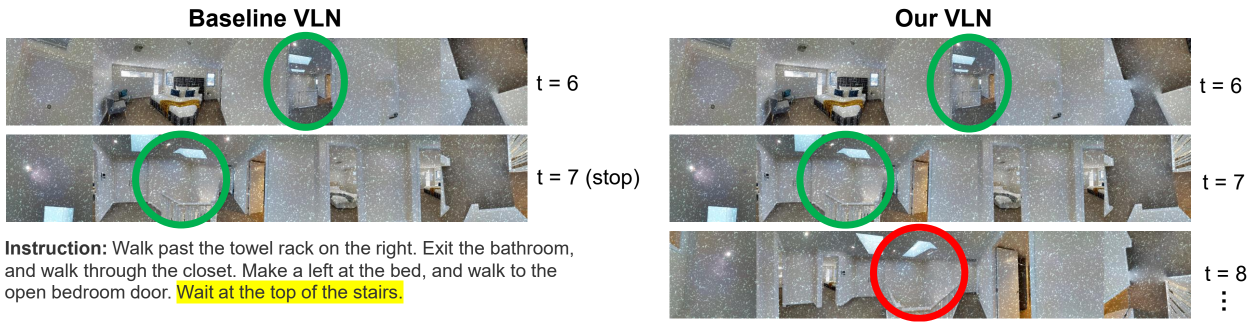
Similarity Metrics	Performance				GFLOPs(↓)
	TL(↓)	OSR(↑)	SR(↑)	SPL(↑)	
Cosine similarity (Ours)	12.87	71.95	60.41	54.50	1917.61
ViT (1st layer activation)	12.89	71.99	60.41	54.59	1966.95
SSIM [57]	12.87	71.95	60.41	54.57	1934.48
FSIM [63]	12.88	71.95	60.45	54.58	1937.73
LPIPS [65]	12.87	71.95	60.49	54.62	1925.15

Table 7. **Impact of employing different similarity metrics in LSH.** We experiment with the HAMT model in R2R.

Table 7 shows our results. Across the board, we observe only a marginal difference between the similarity metrics. We see a performance increase of 0.16–0.22% at the cost of a 2.6% increase in computation. The largest increase in computation comes from obtaining the intermediate activation from ViT. The results indicate that our method is not dependent on the choice of similarity metrics, studied so far in prior work. We also manually analyzed views deemed similar by these metrics, finding most to be identical or having slight variations, e.g., plain walls with lighting differences.

4.4. Robustness to Natural Visual Corruptions

Following recent work [9], we evaluate the robustness of our method's efficiency to practical visual corruptions: Spatter, Defocus Blur, Speckle Noise, Low Lighting, and Motion Blur. Figure 5 shows an example of the most distinct ones. We apply each corruption to the entire validation (unseen) set of R2R, using the corruption framework by Chattopadhyay *et al.* [9]. We set the severity to 3 out of 5, because setting it above 3 causes excessive distortion to the views, which does not reflect the realistic corruptions an agent would encounter.



Instruction: Walk past the towel rack on the right. Exit the bathroom, and walk through the closet. Make a left at the bed, and walk to the open bedroom door. Wait at the top of the stairs.

Figure 4. **Comparison of baseline and our agent trajectories under Spatter corruption.** We demonstrate that our agent fails to stop at the target location, resulting in incorrect navigation (Right), whereas the baseline agent successfully stops as instructed (Left).

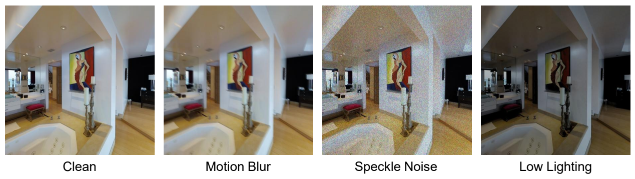


Figure 5. **Examples of the visual corruptions we consider.**

Agent	**Corruption**	**Performance**				
		TL(↓)	**OSR(↑)**	**SR(↑)**	**SPL(↑)**	**GFLOPs(↓)**
HAMT	None	11.53	74.29	66.16	61.49	4763.24
	Spatter	13.30	69.82	58.71	52.91	5227.36
	Defocus Blur	13.87	66.50	55.21	49.32	5383.35
	Speckle Noise	13.60	62.88	51.68	46.02	5345.07
	Low Lighting	12.15	71.31	62.58	57.23	4903.06
	Motion Blur	12.41	68.20	59.13	54.01	4996.64
Ours	None	12.87	71.95	60.41	54.50	1917.61
	Spatter	16.09	67.01	49.04	41.53	2201.19
	Defocus Blur	16.22	63.69	49.21	41.73	2082.57
	Speckle Noise	18.11	61.43	40.87	33.60	2342.67
	Low Lighting	15.27	69.90	52.58	45.33	1516.50
	Motion Blur	14.47	65.47	52.96	46.52	1986.50

Table 8. **Robustness evaluation of baseline HAMT and efficient HAMT under visual corruptions.** We evaluate both models on R2R under clean conditions and five types of visual corruption.

Results. Table 8 summarizes our findings from evaluating the HAMT agent on the R2R benchmark. We first observe that applying our method to a VLN agent reduces its performance and computational savings compared to the original agent. Across the five corruptions, the HAMT agent shows 5.4–21.1% reductions in performance, while our agent undergoes 12.3–31.3% reductions. GFLOPs increase by 2.9–13.0% in HAMT, while we show an increase of 3.6–20.9%. Per corruption, we find that both agents are most resilient to Low Lighting and least robust to Speckle Noise. This aligns with the findings of prior work [9]. HAMT and DUET use visual encoders pre-trained on ImageNet-1K, meaning they inherit the susceptibility of these ImageNet encoders to visual corruptions. This finding highlights the importance of studies on enhancing the robustness of visual encoders to natural corruptions [22, 27, 67], which could improve the robustness across various VLN agents.

Interestingly, while our agent experiences a large drop in SR, the impact on OSR is notably smaller. To uncover why, we manually analyze R2R trajectories. Figure 4 shows a representative path for both agents. Our agent consistently overshoots the target, whereas the baseline stops correctly. The agent should stop at the top of the stairs, but instead moves past them into an adjacent room and continues turning until reaching the step limit. This indicates that our agent can navigate to the target, but struggles to stop in the presence of visual corruption; thus, we hypothesize that our approach mainly affects *recognition* rather than navigation itself.

To test improving the robustness, we apply a median filter (kernel size of 5) to denoise corrupted images. On the most impactful corruption Speckle Noise, we recover SR by 17.9% and reduce GFLOPs by 6.1%. This indicates that denoising is a promising direction for enhancing performance in corrupted environments; as robustness is a separate area of research, we leave further investigation as future work.

5. Conclusion

We propose an input-adaptive inference method to mitigate overthinking in vision-and-language navigation (VLN) and achieve computational efficiency. Unlike the overthinking problem in conventional domains, such as object recognition or natural language comprehension, addressing overthinking in VLN presents three unique challenges: (1) How can we leverage spatial locality in views observed by an agent at a navigation step? (2) How can we reduce temporal redundancy across the agent's navigation steps? (3) How can we use the mechanisms designed to address the two challenges to adaptively set early-exit thresholds of an existing method? We present three novel techniques to address them individually. In our evaluation, we demonstrate a 2–7.5× reduction in computations while preserving performance across seven VLN benchmarks. Moreover, we assess the robustness of our approach under various visual corruptions that may occur in practice, and identify challenges to address for future work. We hope this work inspires future research on developing efficient (and robust) VLN algorithms and promote their widespread adoption in real-world settings.

Acknowledgment

We thank the anonymous reviewers for their valuable feedback. This work is partially supported by the Samsung Global Research Outreach 2024 program. The findings and conclusions in this work are those of the author(s) and do not necessarily represent the views of the funding agency.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 5, 12, 14

[2] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021. 18

[3] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008. 5, 13

[4] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019. 17

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 404–417. Springer, 2006. 7, 16, 17

[6] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 696–697, 2020. 17

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2, 12

[8] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002. 5, 13

[9] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Aniruddha Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15691–15700, 2021. 7, 8

[10] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847, 2021. 1, 2, 5, 6, 12, 14

[11] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16537–16547, 2022. 1, 2, 6, 12

[12] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 17

[13] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019. 17

[14] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 5

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. 6

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 6, 12

[17] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019. 17

[18] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16091–16101, 2023. 17

[19] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1039–1048, 2017. 2

[20] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *Advances in neural information processing systems*, 31, 2018. 2

[21] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. Airbert: In-domain pretraining for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1634–1643, 2021. 1

[22] Yong Guo, David Stutz, and Bernt Schiele. Improving robustness of vision transformers by reducing sensitivity to patch corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4108–4118, 2023. 8

[23] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 17

[24] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015. 17

[25] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13137–13146, 2020. 1

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 12

[27] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 8

[28] Duc NM Hoang and Shiwei Liu. Revisiting pruning at initialization through the lens of ramanujan graph. *ICLR 2023*, 2023. 17

[29] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1643–1653, 2021. 1, 2

[30] Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15439–15449, 2022. 5

[31] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. 1, 2, 4, 6

[32] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 17

[33] Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. A new path: Scaling vision-and-language navigation with synthetic instructions and imitation learning, 2023. 2

[34] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR, 2019. 1, 2, 6

[35] Jacob Krantz and Stefan Lee. Sim-2-sim transfer for vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 588–603. Springer, 2022. 1, 5, 6, 7, 12

[36] Jacob Krantz, Erik Wijmans, Arjun Majundar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision and language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 5, 6, 12

[37] Jialu Li and Mohit Bansal. Improving vision-and-language navigation by generating future-view image semantics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10803–10812, 2023. 2

[38] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 17

[39] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*, 2020. 1, 6

[40] Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. *arXiv preprint arXiv:1810.01875*, 2018. 17

[41] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 7, 16, 17

[42] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 17

[43] Abhinav Moudgil, Arjun Majumdar, Harsh Agrawal, Stefan Lee, and Dhruv Batra. Soat: A scene-and object-aware transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 34:7357–7367, 2021. 1

[44] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 17

[45] Azade Nova, Hanjun Dai, and Dale Schuurmans. Gradient-free structured pruning with unlabeled data. In *International Conference on Machine Learning*, pages 26326–26341. PMLR, 2023. 17

[46] Akhil Perincherry, Jacob Krantz, and Stefan Lee. Do visual imaginations improve vision-and-language navigation agents? In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3846–3855, 2025. 2

[47] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020. 5, 12

[48] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011*

International conference on computer vision, pages 2564–2571. Ieee, 2011. 16, 17

[49] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019. 2

[50] Huixin Sun, Runqi Wang, Yanjing Li, Xianbin Cao, Xiaolong Jiang, Yao Hu, and Baochang Zhang. P4q: Learning to prompt for quantization in visual-language models. *arXiv preprint arXiv:2409.17634*, 2024. 17

[51] S. Tang, Y. Wang, Z. Kong, T. Zhang, Y. Li, C. Ding, Y. Wang, Y. Liang, and D. Xu. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10781–10791, Los Alamitos, CA, USA, 2023. IEEE Computer Society. 1, 2, 3, 6, 12

[52] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 2

[53] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020. 5, 12, 14

[54] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. *arXiv preprint arXiv:1905.11452*, 2019. 17

[55] Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning. *arXiv preprint arXiv:2210.07795*, 2022. 17

[56] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 409–424, 2018. 2

[57] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7, 17

[58] Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12009–12020, 2023. 2

[59] Z. Wang et al. Sim-to-real transfer via 3d feature fields for vision-and-language navigation. *CoRL*, 2024. 18

[60] J. Wasserman et al. Last-mile embodied visual navigation. *CoRL*, 2023. 18

[61] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020. 1, 2, 6

[62] Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024. 2

[63] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. Fsim: A feature similarity index for image quality assessment. *IEEE transactions on Image Processing*, 20(8):2378–2386, 2011. 7, 17

[64] Q. Zhang et al. Humanoidpano: Hybrid spherical panoramic-lidar cross-modal perception for humanoid robots. *arXiv*, 2025. 18

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7, 17

[66] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. Soon: Scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12689–12699, 2021. 5, 12, 14

[67] Sicheng Zhu, Bang An, Furong Huang, and Sanghyun Hong. Learning unforeseen robustness from out-of-distribution data using equivariant domain translator. In *Proceedings of the 40th International Conference on Machine Learning*, pages 42915–42937. PMLR, 2023. 8

[68] S. Zhu et al. Vigor: Cross-view image geo-localization beyond one-to-one retrieval. *CVPR*, 2021. 18